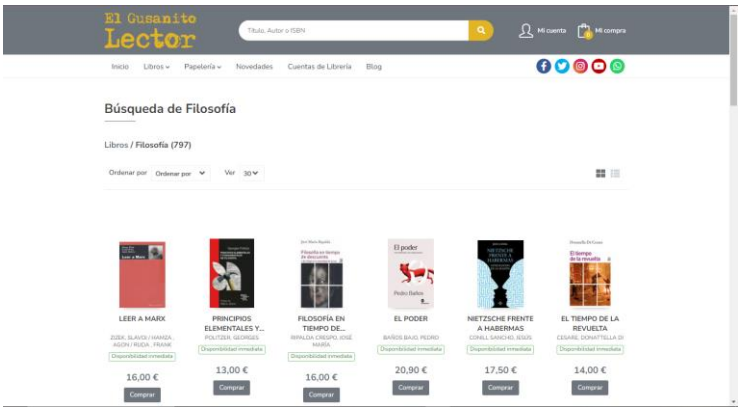
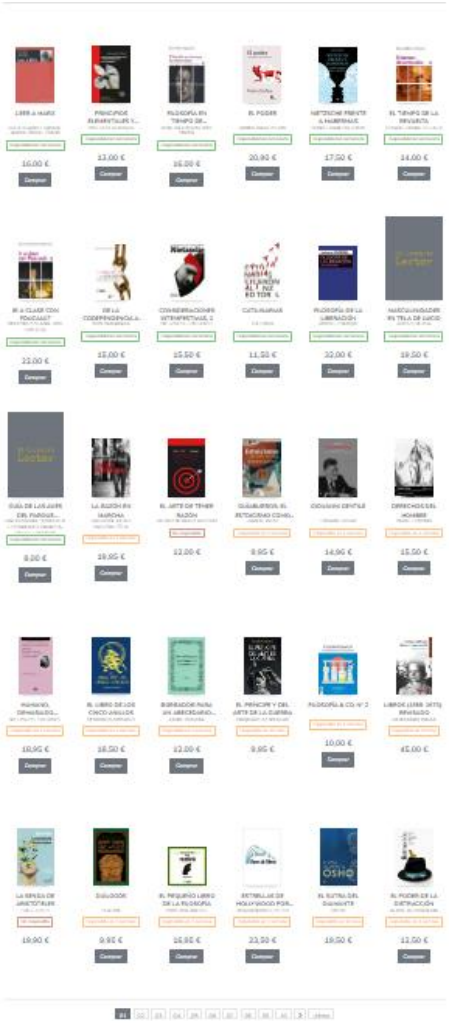
Scrapping el Gusanito lector

La página que he escogido es la de la sección de filosofía de la librería El gusanito lector.

https://www.elgusanitolector.com/libros-de/filosofia-26/





En la parte donde están los libros, están dispuestos en una lista desordenada (ul) y cada libro está en un “il”.

En total tenemos 27 páginas iguales, con 30 libros dispuestos en filas de 6 libros y 5 columnas.

La imagen de la izquierda son todos los libros de la primera página.

Primeramente he importado las librerías reuqest y bs4, para poder hacer el scrapping, luego he creado una plantilla de que información queremos guardar de cada libro.

```
import requests
import bs4

libreria = {
    "portada": "",
    "nombre": "",
    "autor": "",
    "precio": 0.00,
    "descrippcion": "",
    "ISBN": ""
}
```

Ahora creamos un método para obtener la información de toda la página web.

```
def obtener_conrtenido_pagina_web(url):

    #hacemos petición a la url de la página web para obtener su HTML
    pagina_html = requests.get(url)

    #Analizamos el HTML obtenido con bs4
    soup = bs4.BeautifulSoup(pagina_html.content, "html.parser")

    #Devolvemos la sopa de bs4
    return soup
```

Y empezamos con la extracción de los datos.

```
def extracxion_de_datos():
```

Ahora determinamos la variable donde van a estar los libros.

```
      todos_libros = []
```

Ahora llamamos al método anterior por cada página de libros con un “for”. Luego convierto el número de la página en texto (le sumamos 1 al número, porque como empieza a contar por el 0 y no hay página 0), para poder meterlo en el enlace que le pasamos al método que extrae el contenido.

```
#recorremos todas las páginas posibles de la página web
for i in range(27):
    #combertimos la i a string para que se meta en el texto
    i = str(i+1)
    #cogemos la sopa del anterior método
    soup
=obtener_conrtenido_pagina_web(str("https://www.elgusanitolector.com/libros-
de/filosofia-
26/?pagSel="+i+"&cuantos=30&orden=prioridad%2C+fecha_edicion+desc&codMateria=26&t
ipoArticulo=L"))
```

Ahora precisamos la búsqueda y nos vamos a buscar los “li” donde están los libros.

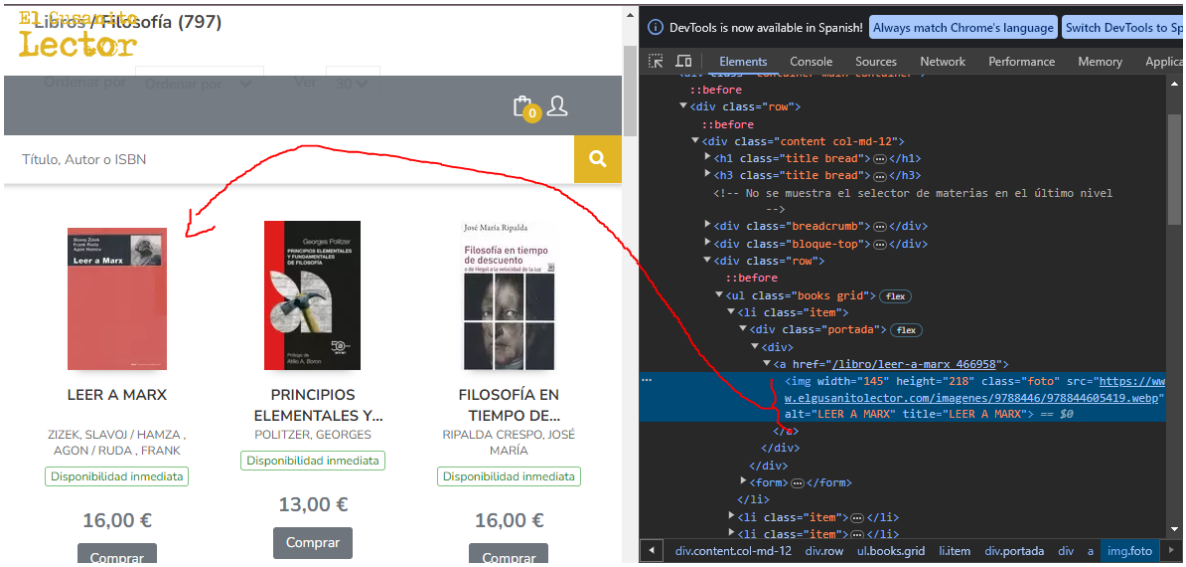
```
# Crear variable para almacenar los elementos
elemento_principal = soup.find("ul", {"class": "books grid"})
#buscamos todos los elementos que está dentro del ul
elementos = elemento_principal.find_all("li")
```

Ahora recorremos todos los “li” para ir extrayendo la información.

```
# recorremos los elementos de la etiqueta y sacamos
for elemento in elementos:
```

Empezamos por la portada que es la única imagen

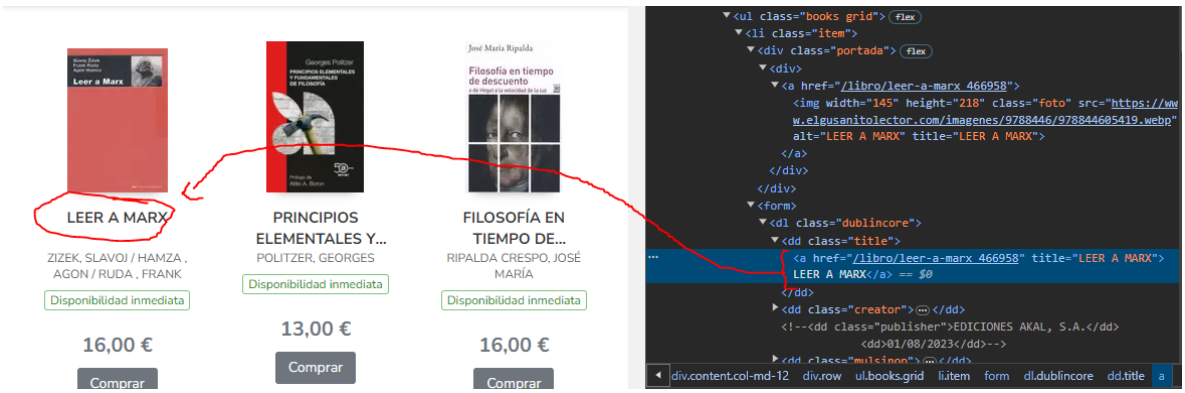
```
    #primero sacamos la portada
imagen_url = elemento.find("img")["src"]
```



Ahora el nombre del libro.

```
#el nombre del libro
nombre = elemento.find("img")["alt"] #el título está en una parte del html que se repite, pero en el alt de la imagen aparece el título
```

Pero se nos presenta un problema, el nombre en el código está en un “a”, al igual que la imagen.



Pero si nos fijamos en el “alt” de la imagen está el título, así que solo necesitamos volver a buscar la imagen, pero en vez del “src”, el “alt”.

El autor también es un dato importante para guardar.

En este caso no nos ha dado mucho problema porque está recogido en un lugar que no se repite.

```
#El autor está mal puesto, así que hay que hacerle unos cambios para que aparezca
bonito
#hacemos la búsqueda
autor = elemento.find("dd", {"class":"creator"}).text[:-2].split("/")
#creamos una variable para poner los autores bonitos
autores = ""
#contamos el número de autores
contador = 1
#recorremos los autores y modificamos lo que no nos gusta
for i in autor:
    contador += 1
    #cuando llegue al último autor pone una "y" en vez de una ","
    if  contador == len(autor):
        autores += i.strip(" ").capitalize() + " y "
    else:
        autores += i.strip(" ").capitalize() + ", "
```

Con el problema del autor es que pueden aparecer varios y así de mal:

```
'          ZIZEK, SLAVOJ      /      HAMZA , AGON      /      RUDA , FRANK  \n'
```

Para ello he nececitado separar los autores y meterlos en una lista, luego uno por uno los pongo bonitos y la primera letra en mayúsculas, también los separo bien con su "," y al final su "y".

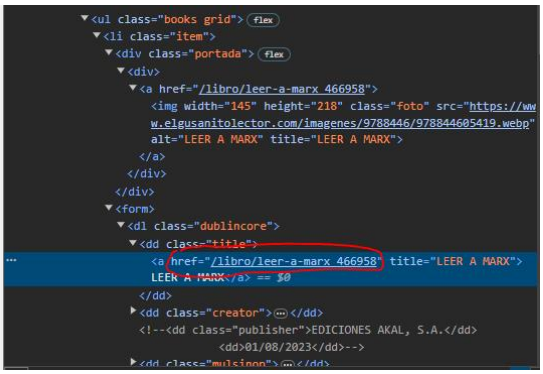
El precio es mucho más facil, ya que es un número y está en un lugar que tampoco se repite.

```
precio = float(elemento.find("strong").text[:-2].replace(",","."))
```

Ahora nos metemos en fregado al buscar el ISBN y la descrippción del libro.

```
#nos metemos en cada sección de cada libro, para ver los detalles de cada libro
detalles_libro =
obtener_conrtenido_pagina_web("https://www.elgusanitolector.com/"+elemento.find("a")["h
ref"])
```

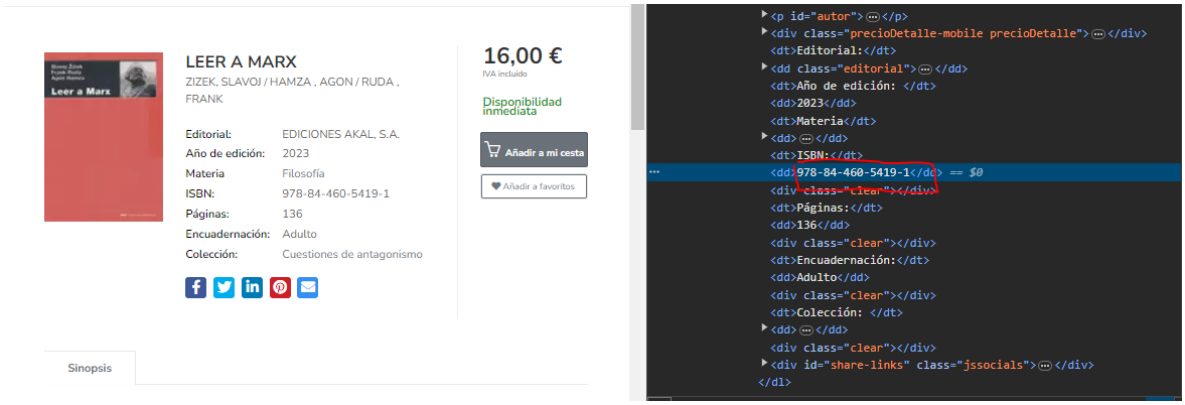
En la página web no está el enlace completo, solo está a mitad, pero no pasa nada, lo rellenamos manualmente escribiendo la parte que falta.



Empezamos a buscar el ISBN de los libros.

```
#primero vemos su ISBN, pero como va variando en la posición las vamos recorriendo y
vemos si cumple los requitos para ponerlo
for i in range(len(detalles_libro.find_all("dd"))):
    if es_un_ISBN(detalles_libro.find_all("dd")[i].text):
        isbn = detalles_libro.find_all("dd")[i].text
```

El problema del ISBN es que en cada sub página está en una posición diferente dentro de una lista de "dd".



Como vemos en los dos casos buscamos el ISBN, pero en los dos libro están en posiciones distintas, así que necesitamos un “for” que vaya comprobando cada “dd” si es un ISBN o no, para ello llamamos a un método que creé arriba del todo con las condiciones (es un número entero y tiene 13 dígitos).

```
def es_un_ISBN(isbn):#creamos un método que comprueba si es un isbm

    isbn = isbn.replace("-","") #le quitamos lo que no queremos, poniéndolo en el
formato deseado
    #todos los isbn tiernen 13 caracteres y son númericos
    if len(isbn) == 13 and es_numero_entero(isbn) == True:
        return True
    else:
        return False
es_un_ISBN("pepe")
```

Pero para comprobar si es un número entero he creado otro método.

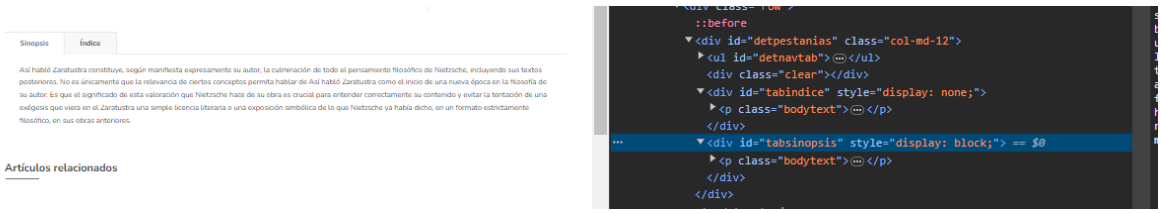
```
def es_numero_entero (numero): #creamos un método para ver si es entero el número

    es_entero = False #variable que dice si es verdad que es entero

    try:  # con el try podemos comprobar si da error el código que le ponemos dentro
        int(numero)
        # Si no hay error, significa que lo que ha escrito el jugador es un número
        es_entero = True
    # Si hay un error, significa que lo que ha escrito el jugador no es un número
    except ValueError:
        es_entero = False

    return es_entero
```

El último dato que busco es la sinopsis o descripción del libro.

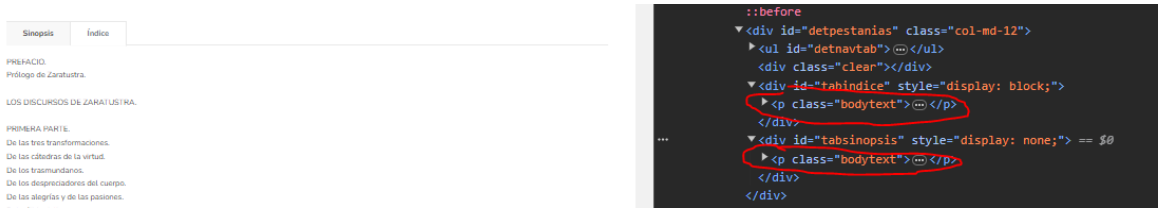
```
#busacmos la sinopsis del libro, pero como algunas no tienen comprobamos si este tiene
o no
if detalles_libro.find("p", {"class": "bodytext"}) == None:
    descrippcion = "no tiene descrippción"
else:
```

```
    descrippcion = detalles_libro.find ("div", {"id":"tabsinopsis"}).find("p", {"class": "bodytext"}).text
```

El problema de la descripción es que puede estar o no, para ello ponemos un condicional para que compruebe si efectivamente hay descripción.



Si nos damos cuenta aparte de la sinopsis está el índice, en el caso este ambos están en un “p”.



Por eso cuando veo que hay descripción busco el id que tiene solo la sinopsis y luego busco el “p”.

Por último vamos creando copias de la plantilla y vamos insertando los datos en la lista con todos los libros.

```
        #vamos insertando a la lista con los libros los diccionarios con las características de cada libro
        nuevo_libro = libreria.copy()
        nuevo_libro["portada"] = imagen_url
        nuevo_libro["nombre"] = nombre
        nuevo_libro["autor"] = autores
        nuevo_libro["precio"] = precio
        nuevo_libro["descrippcion"] = descrippcion
        nuevo_libro["ISBN"] = isbn
        todos_libros.append(nuevo_libro)

return todos_libros
```